**Progress Report**

**- Increment 1 -**

**Group #19**

# Team Members

Dylan McClure

* + **fsuid**: dmm18dk
  + **github**: dylnmc

Julian Sweatt

* + **fsuid**: js15f
  + **github**: juliansweatt

Caleb Smith

* + **fsuid**: cas16w
  + **github**: ninjanole5558

Lucas Zavalía

* + **fsuid**: lrz17
  + **github**: lzavalia

Michael Heron

* + **fsuid**: mph12e
  + **github**: mikeheroncode

1. **Project Title and Description**

Plethora Py:

Plethora Py resembles your typical arcade-style gaming suite. It is a collection of childhood games for the people who grew up playing board games. These digital versions allow a person to take their favorite games with them as long as they have a computer or a Raspberry Pi. Try your hand at some of your favorite childhood games!

1. **Accomplishments and overall project status during this increment**

Each person continued to develop Game 1 and started to implement Game 2. Game 1 for each person was also integrated with the plethora UI and API. Some of the first games still needed some development but everyone besides Dylan who is working on chess and the plethora API and UI have started Game 2 and made some progress while polishing up Game 1. Overall, the project is a little more integrated and polished now since all of the games now have been merged together in master.

Dylan: I continued to develop chess and added move validation, which occupied most of my time. Now, move validation handles all chess moves edge cases like en passant and castling with the exception of handling check, validating discovered check, and detecting check mate. I also integrated chess with the plethoraAPI and vastly improved the user interface for the board. For example, ghost pieces are now used and allow the user to drag a piece to a square in a clean fashion. I also updated the plethora UI element, UIButton, to use callback functions to make button callbacks easier. I also started to work on some other UI elements along with some other core plethora API bug fixes.

Michael: During this increment I did a bit of work refactoring Multisnake to fit the new API and started both poker and blackjack. I began development on poker and blackjack at the same time because they shared a lot of the same parts. For blackjack I have fully implemented the game logic and only have small bug fixes as well as a few GUI tweaks left before it will be finished. Poker’s game logic is mostly implemented aside from a few edge cases determining tiebreakers. I started to work on the GUI but haven’t completed it.

Julian: During this increment, I made finishing touches to my first game, Connect 4, and started my second game, Bomberman. The majority of the work for bomberman so far has been creating a spritesheet and animated graphics library (which can be used by other games in the future). I have established a randmly generated 2d map and a player sprite capable of movement with a death animation.

Lucas: During this increment I originally got to work with the next game, 2048, immediately after the first increment. For 2048 I plan on using matrix algebra to represent motions that are possible. Soon after some of the initial considerations for 2048, I shifted focus trying to get my old game, tetris, working on the API that Dylan developed for this project.

Caleb: integrated tictactoe with plethora API and started checkers.

1. **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

Michael: When I originally designed multisnake I was handling movement completely differently than I am going to handle it now with the API. I don’t think either is more or less effective for performance but the using the API has been fantastic and makes integration very simple. I also had a hard getting the correct value for hands in blackjack when there were aces in the hand becuase an ace can represent two different values based on the other cards in your hand. I also had a difficult time and had to order check for determining winning hands in a very specific order so when there are hands that are similar in content or are just similar in overall hand value the comparisons are done correctly.

Julian: Working with sprites was considerably more difficult than drawing circles in my previous game. I hadnt expected for the sprite sheets and animations to be so complicated, which set back my intended progress for Bomberman. However, I do not think it will be too difficult to complete in the next increment, as the game is about half complete.

Lucas: I had significant challenges updating tetris to the new API and it require not one but two whole rewrites before I had a workable product. To this end I realize that a lot of issues that I was having had to do with misunderstandings I had about programming with Python. For 2048 My most pressing issue will be getting the mechanics to work properly. In particular I need to fix the way random tiles are chosen each time an event occurs. I will also need to merge 2048 with the API but this will be easier to do since 2048 is in its early stages and I have a much better understanding of the API.

Dylan: Move validation was by far the hardest thing to do. Luckily, I have played chess recreationally, so it was easy for me to reason about and implement the validation in a couple of days. Regarding changes in scope, I don’t think I will be implementing more games than chess simply because the chess engine and API/UI will take up the rest of my time.

Caleb: It took a bit of work, but I was able to get integrated with the plethora API. Also, since I am somewhat new to python, implementing checkers has been difficult. There are no changes in plan for my contribution.

1. **Team Member Contribution for this increment**

* Caleb Smith:
  1. progress report
     + 1. Accomplishments: “Caleb:” section
       2. Contributions: “Caleb Smith” section
  2. requirements
     + helped with overview
     + wrote functional requirements overall #3
  3. implementation and testing
     + helped write Execution-based Non-Functional Testing
  4. source code
     + tictactoe integrated with plethora API
     + started checkers
  5. video
     + intro
     + screen recording for tictactoe and checkers
* Lucas Zavalía:
  1. progress report
     + 1. Accomplishments: “Lucas:” section
       2. Contributions: “Lucas Zavalía” section
  2. requirements
     + added tetris user case diagram
  3. implementation and testing
     + added to 3. Execution-based Functional Testing
  4. source code
     + tetris
     + 2048
  5. video
     + helped with intro
     + screen recording for tetris and 2048
* Dylan McClure
  1. progress report
     + 1. Accomplishments: “Dylan:” section
       2. Contributions: “Dylan McClure” section
     + created stubs for everyone to easily work
  2. requirements
     + rewrote overview
     + rewrote functional requirements
     + created stubs for everyone to easily work
  3. implementation and testing
     + updated Execution-based Non-Functional Testing
  4. source code
     + chess: refactored chess, implemented move validation, and upgraded UI
     + API: minor change to allow for easy callbacks for buttons
     + UI: started to add new UI elements including UIDiaglog
  5. video
     + intro
     + chess, plethoraAPI/UI screen recording
     + edited video
* Julian Sweatt:
  1. progress report
     + 1. Accomplishments: “Julian:” section
       2. Contributions: “Julian Sweatt” section
  2. requirements
     + Wrote the updated functional requirements of connect4 and bomberman
  3. implementation and testing
     + Execution-based Functional Testing, details about cross-platform continuous support
     + Non-Execution-based Testing, added information about mandatory code rewviews (and set up github to enforce it)
  4. source code
     + Connect 4: Finishing touches and increased integration with the API
     + Common: Built common/shared libraries for spritesheets and animated/player controlled graphics
     + Bomberman: Started bomberman by building a dynamically generated map and a moveable player sprite ontop of that map
  5. video
     + Contributed to the intro
     + Screen recorded a demo of my progress made during this increment
* Michael Heron:
  1. progress report
     + 1. Accomplishments: “Michael:” section
       2. Contributions: “Michael Heron” section
       3. Plans: “Michael” section
  2. requirements
     + Use case diagram
     + Non-Functional Requirements
  3. implementation and testing
     + Execution based testing
  4. source code
     + MultiSnake: began API integration
     + Blackjack: Mostly completed development and fully integrated into UI
     + Poker: partially implemented game logic and partially implemented UI
  5. video
     + Contributed to the intro
     + Recorded demo for Blackjack

1. **Plans for the next increment**
2. Michael: I will finish Blackjack, Poker and MultiSnake. Blackjack is almost compltely finished with only minor tweaks left. Poker needs to be fully integrated into the api and have end game conditions as well as conditions to start a new game. I also plan to add ai oppentents who will all wager against the player to make it feel more lively. Multisnake will also need to be completely integrated into the api as well as adding support for a third and fourth player. I also plan on doing adding in speed and snake size settings.

Julian: I plan to finish Bomberman, specifically, add a second player, colision handling with the map/enivronment and the player, and the actual bomb mechanic. I also plan to integrate my existing games with the UI elments that Dylan created this increment.

Caleb: Finish implementing checkers and start on my next game. Also, I will continue using the API for my games, and will also see if I can work on integrating some new UI elements for tictactoe (for example, the buttons)

Lucas: I still need to make tetris work properly. Luckily, this will be my hardest game, as 2048 will be much easier. So, I plan on finishing tetris and 2048 for the next increment.

Dylan: finish chess engine; add UI elements such as Dialog, Menu, Input Text, etc; improve API to handle UI elements, call onstart() for each game, etc; coordinate with team mates to standardize UI and utilize UI elements once they are more stable.

1. **https://youtu.be/7Dj6tPjJulE**